LA-UR -85-3781

CONF-851009--10

LA-UR--85-3781

DE86 002437

TITLE: # A 32-BIT FASTBUS COMPUTER

AUTHOR(S): J. M. Blossom, J. P. Hong, and R. G. Kellner

SUBMITTED TO: IEEE 1985 Nuclear Science Symposium,
San Francisco, CA, October 23-25, 1985

Los Alamos Los Alamos National Laboratory
Los Alamos, New Mexico 87545

A 32-BIT FASTBUS COMPUTEP

J.M. Blossom, J. L. Hong and R.G. Kellner
Los Alamos National Laboratory
E-10 Data Systems, MS K488
Los Alamos, NM 87545

Abstract

Los Alamos National Laboratory is building a 32-bit FASTBUS computer using the NATIONAL SEMICONDUCTOR 32032 central processing unit (CPU) and containing 16 million bytes of memory. The board can act both as a FASTBUS master and as a FASTBUS slave. It contains a custom direct memory access (DMA) channel which can perform 80 million bytes per second block transfers across the FASTBUS.

Overview

The FB32000 FASTBUS computer has been designed to optimize high-speed block transfers across the FASTBUS, along with providing on-line data reduction and functional control. A 32-bit UNIX* engine, the NS32032, is resident on a separate execution bus, freeing the FASTBUS for other operations while data reduction proceeds on the FB32000. Block transfer of data is facilitated by a custom DMA channel which assembles and moves heterogeneous linked lists of data blocks. A 16 million byte memory allows storage of large data tables without frequent disk accesses. An Ethernet port connects the FB32000 to external devices.

In order to support 32-bit data calculations and efficient data movement within the FASTBUS environment, a 32-bit microcomputer with virtual memory management was chosen as the main processing element. The National Semiconductor NS32032 family with hardware floating point, interrupt control, memory management and DMA controller provides all necessary functions in a complete chip set. The NS32032 has a highly orthogonal instruction set which eases the software development task. The CPU and support chips reside on a local execution bus, reserving the FASTBUS for inter-device block transfers via the custom DMA channel.

A custom high-speed DMA channel between the FB32000 memory and the FASTBUS is intended for multiple block transfers of possibly non-contiguous data fields located anywhere in memory. Block size can be heterogeneous and may vary from one 32-bit word to the entire four million word memory. A hardware pointer processor allows stripping headers or trailers from data blocks and storing them in a separate area in memory without intervention by the software.

The FASTBUS interface is based on programmable array logic (PALs) and ASKOM FMA601 gate array chips. The interface supports master/slave operation, parity generation and checking, and single or block mode transfers. The FB32000 can be addressed in either geographical or logical address space.[1]

Detailed Description

The National Semiconductor NS32032 and its supporting chip set furnish high-level language support through a very orthogonal instruction set, hardware memory management and a floating point slave processor. The CPU has eight 32-bit internal registers connected by 32-bit internal and external data paths. The 24-bit external address port accesses 16 million bytes of random access memory (RAM). When clocked at 10 MHz, the NS32032 executes 1 million instructions per second.

A floating point slave processor, the NS32081, performs single and double precision IEEE standard P754 binary floating point arithmetic. The NS32081 features eight 32-bit registers, and performs a double precision floating point multiply in 6.2 μsec when clocked at 10 MHz.

Interrupt control and counter-timer functions are combined in the NS32202 interrupt control unit (ICU). The ICU contains two 16-bit counters, which may be cascaded, and responds to 16 maskable prioritized interrupts. An interrupt vector is generated by the ICU to speed processing of external events.

The NS32082 memory management unit (MMU) incorporates a 32 entry on-chip translation cache which is automatically updated from page tables in memory. Address translation is performed on chip, with 24-bit logical to 24-bit physical mapping of 512-byte pages. The MMU also supports program development with hardware breakpoint and program trace-back registers.

Peripheral devices resident on the internal bus of the FB32000 have access to DMA provided by the NS32203 DMA control chip. The NS32203 is functionally separate from the high-speed custom DMA channel to the FASTBUS, and operates at up to 5 million bytes per second transfer rates. Four 16-bit I/O channels can format data from 8-bit peripherals into 16-bit words for efficient utilization of bus resources.

Time stamping of data files and an interrupt alarm function are realized with the MM58167 real time clock. Battery backup prevents loss of time synchronization during power down.

Software development, file system access, process control, simple maintenance, and local area networking are possible through an on-board Ethernet port. Ethernet is implemented with the Advanced Micro Devices AM7990-AM7992 chip set utilizing a buffered DMA port to memory.

A large memory resides on the local FB32000 execution bus allowing storage and processing of very large data tables or keeping a large program resident in memory without the necessity of frequent disk accesses. The memory is divided into 128K bytes of read only memory (ROM), and up to 16 million bytes of RAM. The RAM section is divided into four banks of four tiers each, with each tier comprising four single in-line surface mounted memory modules of nine RAM chips each. The board is configurable to use either 64K bit or 256K bit chips and may have from one to four tiers populated at a time, resulting in memory configurations of from 1 million bytes to 16 million bytes of

*UNIX is a trademark of AT&T

RAM. The four memory banks are interleaved at the word level in order to speed block transfers by read-ahead and write-behind of data. Parity is generated and checked at the memory port and stored byte-wise in RAM. Word parity is also generated and checked at the FASTBUS interface.

Each memory bank is controlled by a synchronous state machine implemented in Texas Instruments B-PALs. These state machines handle timing and data latch control as well as controlling the bank address counter in block transfer mode. The banks cycle independently and are run in page mode at all times. Page mode timing allows many column accesses per row access, greatly improving access time and minimizing operating power. An address comparator, located in each bank, detects a change in the row address and causes a row address cycle whenever necessary. In block transfer mode, all addresses increase monotonically, causing a row address cycle once every 512 addresses, and additionally once every 9.5 μ sec due to the row address strobe timing out. In a normal access mode, program and data locality tends to keep row address cycles to a minimum.

Refresh control is synchronized across all tiers and banks by a PAL-based state machine that allows the banks to refresh as soon as the current data cycle is finished. In burst mode, this results in a refresh ripple across the banks since each bank is skewed by one FASTBUS transfer time.

A custom DMA channel between the FASTBUS and the FB32000 memory controls burst-mode data transfers on and off the FASTBUS. The DMA channel operates at burst speeds up to 80 million bytes per second and incorporates several features useful in file transfer. A pointer register file of 1K 24-bit address pointer pairs is accessible by the CPU in both read and write mode. This register file contains pointers to starting and stopping addresses of the data blocks to be moved. The pointer pairs can point to any size block from one word to 4 million words, or the entire contents of RAM memory. As no sequence or set size is imposed on the pointer list, data may be moved in heterogeneously sized blocks located anywhere in memory, and one data block may overlap another data block. The DMA control register contains a data source bit which may be set to clear an array to either 00000000 or FFFFFFFF (hex) at DMA transfer speed. This feature may be used to initialize an array to all ones or all zeros at 80 million bytes per second.

Design of the memory, interleaver, and DMA subsystem was accomplished with a register transfer hierarchical approach and several automated engineering aids. The schematic capture was done on VALID SCALD design stations, state machine designs were compiled using the Berkeley VLSI Tool Sets' PEG program, and PALS were generated by PALASM and CUPL. The main data paths, registers, and special functions such as parity generation and address comparison were blocked out at the system level, and then expanded iteratively without regard to register control or sequencing. An algorithmic description was then developed for data path control, and portions of the algorithm translated into PEG source language. At this point critical timing paths were identified, and register implementations were modified where necessary. PEG, an acronym for Programmable array logic Equation Generator, outputs Boolean equations for a Moore model state machine in which outputs are strict functions of the state variables, given a suitable algorithm as input. The output is suitable for implementation of the logic design in a custom VLSI chip, so a filter, PEGASM, was developed by members of our group to transform the output into PALASM format. After the PEG source is compiled and translated into PALASM, a PAL device may be programmed. The result is an implementation of the control function with off-the-shelf PAL devices. CMOS UV-erasable PALS were used during development to allow cost effective iteration of the design. Some portions of the control logic, e.g. the pipeline handshake between the interleaver and the FASTBUS data interface, were not possible to design using PEG because of asynchronous operation, speed constraints, or both. In these instances the design was simulated on Cericor CAE software running on a Sun workstation, or in some instances on the VALID logic design stations. Major subblocks of the design were prototyped, including the CPU, memory banks, Ethernet interface, and FASTBUS interface.

Certain unexpected timing constraints were uncovered as the design progressed, such as the surprising result that, due to pipeline delays, the 80 million bytes per second memory interface will just keep up with no-wait-state operation of a 10 MHz 32032 CPU fetching one operand in 400-500 ns. In order to satisfy the NS32032's memory fetch timing, 120 ns DRAMs were used, and a custom RAM controller was designed to take full advantage of access times. A state machine approach was used to design the RAM controller, which had to handle page mode timing control, address source multiplexing, and random and block mode accesses. A 20 MHz clock speed was used to cut timing penalties associated with the quantized timing of the state machine. Glitch-free operation necessitated a registered variable approach, forcing the division of the RAM controller into four PAL devices. The design was implemented in Texas Instruments TIBPAL16R8 logic devices, and ran glitch-free out to 40 MHz.

The DMA channel consists of a 22-bit address counter, a 2K word 24-bit address stack, a stack pointer counter, a control register, various data and address paths, and a state machine to control DMA operation. A 20-bit address counter located in each memory bank is also associated with burst mode data transfer through the DMA channel. The memory is run with a read-ahead/write-behind scheme to take full advantage of the inherent pipelining of the interleaved memory design. Since this requires the memory banks to be accessing different addresses at the same time, separate address counters were used for each bank.

Included below is a sample PEG source file used to implement the refresh controller. Comments appear as a double dash (--) followed by text to the end of the line. The actual program starts with the "INPUTS" key word and includes constructs for state sequence control and output variable assertion. The control statements allow n-way branching (CASE, GOTO), idling (LOOP), and control based on single and multiple input variables (CASE, IF-THEN).

This is the peg source file for the FASTBUS multi-bank refresh controller. The controller handles the following functions:

(1) It detects the RFCK output from the National Semiconductor DP84300 refresh timer, and generates 4 independent refresh requests to the 4 memory banks in the FASTBUS card.

(2) It independently resets each RFRQn output on receipt of RFADn(L) from each bank.

(3) On receiving all four RFADn(L) signals from the memory banks, it generates an INCRFAD (increment refresh address) signal to the refresh address counter, and then loops back to waiting for RFCK.

```
--SIGNAL LIST:

--RESET:  This signal clears all registers. (low
          --true)

--RFCK:  Input (high true) from DP84300. Signals
         --time for refresh.

--RFAD3..RFAD0:  Input (low true) from memory
                 --banks 3..0. Memory banks
                 --generate this signal while
                 --performing a refresh cycle.
                 --Used to detect refresh
                 --complete.

--RFRQ3..0:  Output (high true) signals bank(n)
             --to perform refresh.

--INCRFAD:  Output (high true). Used to
            --increment refresh address
            --counters. Its complement is fed
            --to the DP84300s' REFRESH input
            --to indicate refresh complete
            --across all banks.

INPUTS:  RESET RFCK RFAD3 RFAD2 RFAD1 RFAD0;
OUTPUTS:  RFRQ3 RFRQ2 RFRQ1 RFRQ0 INCRFAD;

zero:  IF NOT RFCK THEN LOOP;
       --this is the reset state.
one:  ASSERT RFRQ3 RFRQ2 RFRQ1 RFRQ0;
      CASE (RFAD3 RFAD2 RFAD1 RFAD0)
      1 1 1 1  →   two;
      ENDCASE  →   LOOP;--wait for all banks
                          --to respond.
two:  ASSERT INCRFAD;--reset all RFRQ lines,
                     --increment refresh count
                     --and handshake to DP84300
three:  ASSERT INCRFAD;
        GOTO zero;
```

GENIX, a National Semiconductor port of Berkeley 4.2 bsd UNIX* to the NS32032, was purchased by the Los Alamos National Laboratory and the source code is being modified to accomplish a UNIX* port to the FB32000.

The FASTBUS interface is divided into four sections: master logic; slave logic; CSRs; and the FASTBUS to local bus interface. Askom FMA601 ECL macrocell chips were used to implement the data path in the FASTBUS to local bus interface. The FMA601 consists of an 8-bit slice of the following functions: FASTBUS drivers; FASTBUS receivers; output latches; parity logic; logical address register and compare; and local bus drivers and receivers. The interface also includes ECL-TTL level translation utilizing 10124 and 10125 buffer chips.

The FB32000 implements CSRs 0, 3, 5, 7, 8, 9, A-F and 100 through 10F. CSR0 contains the device I.D. and 16 control and status bits. CSR3 is the logical address register, and CSR8 is the master arbitration level register. CSR5, CSRs A-F and 100-10F are used for interdevice block transfer parameters. CSR7 is used for storing the broadcast mode case 2 class "N" values in slave mode. CSR9 provides master timing control for diagnostic purposes.

Geographical, logical, and broadcast cases 1-5 primary address cycles are supported by the slave logic. Block transfer, random read/write, and secondary address cycles are fully supported. The slave logic is based on a PAL design by Downing and Pregernig[2].

The master logic is implemented in a finite state machine which coordinates all FASTBUS and local bus operations. The state machine is controlled by a local bus CSR with read/write access by the NS32032.

Acknowledgement

This work was supported by the U.S. Department of Energy.

References

[1] R. Kellner, J.P. Hong, and J.M. Blossom, "A 32-bit Computer For Large Memory Applications On The FASTBUS", IEEE Trans. Nucl. Sci., Vol. NS-32 No.4, August 85.

[2] Downing, R.W. and Pregernig, L., " A Simple FASTBUS Slave Interface Using PALS", IEEE Trans. Nucl. Sci., Vol. NS-32, No.1, February 85.